Luis Mario Ayala Castellanos

ID: 174902

Temas Selectos 3: Uso de la Web Semántica en IoT

Julio Noe Hernández Torres

25/11/2025

1. Introducción

El presente documento describe el desarrollo de un sistema IoT aplicado a un salón de clases, integrando datos sensoriales con tecnologías de la Web Semántica. El objetivo general es demostrar cómo RDF, ontologías como SSN/SOSA y consultas SPARQL permiten estructurar, representar y consultar información de sensores de manera interoperable, estandarizada y semánticamente rica.

Este proyecto sigue el flujo completo requerido por el curso:

1. Generación de datos IoT,
2. Modelado semántico con RDF y SSN/SOSA,
3. Carga en un triplestore GraphDB y
4. Consulta semántica mediante SPARQL usando Python.

2. Fundamentos Teóricos

2.1. Internet de las Cosas (IoT)

El Internet de las Cosas es un paradigma tecnológico donde dispositivos físicos equipados con sensores, actuadores y conectividad recopilan datos del entorno y los envían para su análisis. IoT enfrenta varios desafíos como heterogeneidad, interoperabilidad, escalabilidad y la falta de un modelo de datos unificado que permita integrar diferentes plataformas o dispositivos.

Es aquí donde la Web Semántica aporta valor para resolver problemas de compatibilidad y estandarización.

2.2. Web Semántica

La Web Semántica propone representar información con significado explícito mediante modelos de datos estructurados. Su objetivo es permitir que máquinas puedan interpretar los datos, no solo almacenarlos.

Tecnologías principales:

- **RDF (Resource Description Framework):** modelo basado en triples sujeto–predicado–objeto.
- **OWL (Web Ontology Language):** lenguaje para crear ontologías complejas.
- **SPARQL:** lenguaje de consulta diseñado específicamente para datos RDF.
- **URIs:** identificadores globales únicos que representan entidades del dominio

2.3. Ontologías SSN/SOSA

SOSA (Sensor, Observation, Sample & Actuator) y SSN (Semantic Sensor Network) son las ontologías estándar del W3C para describir sensores y observaciones.

Conceptos clave:

- **sosa:Sensor:** dispositivo que mide una propiedad.
- **sosa:ObservedProperty:** propiedad observada (ej. temperatura).
- **sosa:Observation:** evento individual de medición.
- **sosa:hasResult:** resultado de la medición.
- **sosa:resultTime:** tiempo en que se registró la observación.
- **sosa:hasSimpleResult:** valor numérico del resultado.

Estas ontologías permiten representar de manera formal y estandarizada cualquier sistema IoT.

3. Descripción del Caso de Estudio

El proyecto simula un sistema IoT en un **salón de clases**, equipado con un **sensor de temperatura** y un **sensor de humedad**.
Se generó un dataset correspondiente a una semana completa, con mediciones cada 10 minutos durante las 24 horas del día.

En total, la semana contiene:

- 144 observaciones por día
- 1,008 observaciones por propiedad
- 2,016 observaciones totales (temperatura + humedad)

Este dataset forma la base del modelado semántico posterior.

4. Generación del Dataset Simulado

El dataset fue generado mediante un script en Python que produce valores realistas de temperatura y humedad para un clima controlado de salón.

```python
def simulate_classroom_data():
    # Parámetros de simulación
    start_date = datetime(2025, 10, 1, 0, 0, 0)   # puedes ajustar la fecha de inicio
    end_date = start_date + timedelta(days=7)
    interval = timedelta(minutes=10)

    data = []

    current_time = start_date
    while current_time < end_date:

        # Temperatura simulada (°C)
        # Más caliente al mediodía, más fresco en la madrugada
        base_temp = 22

        # Variación por hora
        hour_variation = (current_time.hour - 12) ** 2
        temp = base_temp + random.uniform(-2, 2) - (hour_variation * 0.05)

        # Entre 18 y 29°C en condiciones normales
        temp = max(18, min(temp, 29))

        # Humedad simulada (%)
        humidity = random.uniform(40, 65)

        data.append({
            "timestamp": current_time.isoformat(),
            "temperature": round(temp, 2),
            "humidity": round(humidity, 2)
        })

        current_time += interval

    return data
```

Fig 1. Script en Python para generar un dataset IoT simulado de temperatura y humedad.

5. Modelado Semántico con RDF y SSN/SOSA

El dataset generado se transformó a formato RDF utilizando la ontología SOSA. Cada observación incluye:

- URI única
- Tipo de observación
- Propiedad observada
- Tiempo
- Resultado
- Valor medido

```python
def csv_to_rdf_turtle(csv_file, ttl_file):
    prefixes = """
@prefix : <http://example.org/iot/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .
@prefix qudt: <http://qudt.org/schema/qudt/> .
@prefix qudt-unit: <http://qudt.org/vocab/unit/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:sensor1 a sosa:Sensor ;
    sosa:observes :Temperature, :Humidity .

:classroom1 a sosa:FeatureOfInterest ;
    sosa:hasProperty :Temperature, :Humidity .

"""

    obs_counter = 1
    ttl_content = prefixes

    with open(csv_file, newline="", encoding="utf-8") as f:
        reader = csv.DictReader(f)

        for row in reader:
            timestamp = row["timestamp"]
            temperature = row["temperature"]
            humidity = row["humidity"]

            # ---------- TEMPERATURE OBSERVATION ----------
            obs_id = f"obs{obs_counter:04d}"
            result_id = f"result{obs_counter:04d}"

            ttl_content += f"""
:{obs_id} a sosa:Observation ;
    sosa:hasFeatureOfInterest :classroom1 ;
    sosa:observedProperty :Temperature ;
    sosa:madeBySensor :sensor1 ;
    sosa:resultTime "{timestamp}"^^xsd:dateTime ;
    sosa:hasResult :{result_id} .

:{result_id} a sosa:Result ;
    sosa:hasSimpleResult "{temperature}"^^xsd:float ;
    qudt:unit qudt-unit:DegreeCelsius .
```

Fig 2. Fragmento del script en Python para la conversión del dataset generado CSV a turtke .ttl

Se utilizó el formato Turtle (.ttl) por su legibilidad.

sensor_data: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
@prefix : <http://example.org/iot/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .
@prefix qudt: <http://qudt.org/schema/qudt/> .
@prefix qudt-unit: <http://qudt.org/vocab/unit/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:sensor1 a sosa:Sensor ;
    sosa:observes :Temperature, :Humidity .

:classroom1 a sosa:FeatureOfInterest ;
    sosa:hasProperty :Temperature, :Humidity .


:obs0001 a sosa:Observation ;
    sosa:hasFeatureOfInterest :classroom1 ;
    sosa:observedProperty :Temperature ;
    sosa:madeBySensor :sensor1 ;
    sosa:resultTime "2025-10-01T00:00:00"^^xsd:dateTime ;
    sosa:hasResult :result0001 .

:result0001 a sosa:Result ;
    sosa:hasSimpleResult "18"^^xsd:float ;
    qudt:unit qudt-unit:DegreeCelsius .

:obs0002 a sosa:Observation ;
    sosa:hasFeatureOfInterest :classroom1 ;
    sosa:observedProperty :Humidity ;
    sosa:madeBySensor :sensor1 ;
    sosa:resultTime "2025-10-01T00:00:00"^^xsd:dateTime ;
    sosa:hasResult :result0002 .

:result0002 a sosa:Result ;
    sosa:hasSimpleResult "58.51"^^xsd:float ;
    qudt:unit qudt-unit:Percent .
```

Fig 3. Fragmento del archivo RDF en formato Turtle, modelado con la ontología SOSA.

6. Carga en GraphDB

Se utilizó GraphDB como triplestore, siguiendo las instrucciones:

1. Crear un repositorio nuevo desde la Workbench.
2. Configurarlo como repositorio graphdb:SailRepository.
3. Cargar el archivo .ttl mediante Python usando SPARQLWrapper.

```python
import requests

def upload_turtle_to_graphdb(repo_id="iot_repo", ttl_file="C:\\Users\\luism\\OneDrive\\Documentos\\A. 9no Semestre\\Temas 3 IOT\\sensor_data.ttl"):
    url = f"http://localhost:7200/repositories/{repo_id}/statements"

    headers = {"Content-Type": "text/turtle"}

    with open(ttl_file, "rb") as f:
        data = f.read()

    response = requests.post(url, data=data, headers=headers)

    if response.status_code == 204:
        print("Archivo Turtle cargado exitosamente.")
    else:
        print("Error al cargar Turtle:", response.text)

upload_turtle_to_graphdb("iot_repo", "sensor_data.ttl")
```

Figura 4. Script en Python utilizado para cargar el grafo RDF en GraphDB.

7. Consultas SPARQL

Una vez cargado el dataset, se ejecutaron consultas desde Python hacia GraphDB usando SPARQLWrapper.
A continuación, se describen las consultas utilizadas en el proyecto.

Se mostraran los resultados de las queries desde el Workbench de GraphDB para mejor visibilidad ya que desde terminal de VS Code puede llegar a ser confuso por el formato

7.1. Obtener cinco observaciones

```
# CONSULTA 1: Obtener 5 observaciones

query_all = """
PREFIX sosa: <http://www.w3.org/ns/sosa/>

SELECT ?obs ?time ?value
WHERE {
    ?obs a sosa:Observation ;
         sosa:resultTime ?time ;
         sosa:hasResult ?r .
    ?r sosa:hasSimpleResult ?value .
}
LIMIT 5
"""
```

Fig 5. Consulta SPARQL de 5 observaciones

Resultado:

Muestra las primeras cinco observaciones del dataset, incluyendo su tiempo y valor.

	obs	time	value
1	:obs0001	"2025-10-01T00:00:00"^^xsd:dateTime	"18"^^xsd:float
2	:obs0002	"2025-10-01T00:00:00"^^xsd:dateTime	"58.51"^^xsd:float
3	:obs0003	"2025-10-01T00:10:00"^^xsd:dateTime	"18"^^xsd:float
4	:obs0004	"2025-10-01T00:10:00"^^xsd:dateTime	"58.59"^^xsd:float
5	:obs0005	"2025-10-01T00:20:00"^^xsd:dateTime	"18"^^xsd:float

Fig 6. Resultado de ejecutar la consulta SPARQL para obtener cinco observaciones.

7.2. Temperatura máxima registrada

```
query_max_temp = """
PREFIX : <http://example.org/iot/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT (MAX(xsd:float(?value)) AS ?maxTemp)
WHERE {
    ?obs sosa:observedProperty :Temperature ;
         sosa:hasResult ?r .
    ?r sosa:hasSimpleResult ?value .
}
"""
```

Fig 7. Consulta SPARQL de temperatura máxima

Nos muestra la temperatura máxima que alcanzo el salón de clases en la semana de registro

Resultado:

	maxTemp
1	"23.95"^^xsd:float

Fig 8. Resultado de Consulta SPARQL para obtener la temperatura máxima durante la semana.

7.3. Temperatura promedio

```
query_avg_temp = """
PREFIX : <http://example.org/iot/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT (AVG(xsd:float(?value)) AS ?avgTemp)
WHERE {
    ?obs sosa:observedProperty :Temperature ;
         sosa:hasResult ?r .
    ?r sosa:hasSimpleResult ?value .
}
"""
```

Fig 8. Consulta SPARQL de temperatura promedio

Nos muestra la temperatura promedio del salón de clases durante la semana.

Resultado:

	avgTemp
1	"20.030788"^^xsd:float

Fig 9. Resultado de Consulta SPARQL de temperatura promedio

7.4. Humedad mínima

```
PREFIX : <http://example.org/iot/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT (MIN(xsd:float(?value)) AS ?minHumidity)
WHERE {
    ?obs sosa:observedProperty :Humidity ;
         sosa:hasResult ?r .
    ?r sosa:hasSimpleResult ?value .
}
"""
```

Fig 10. Consulta SPARQL de humedad mínima registrada

La cual nos devuelve la humedad mínima que se registro con el dispositivo en la semana

Resultado:

	minHumidity
1	"40.02"^^xsd:float

Fig 11. Resultado de Consulta SPARQL de humedad mínima registrada

7.5. Consultas por rango de fechas del 2 de octubre al 3 de octubre

```
PREFIX : <http://example.org/iot/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?obs ?time ?value
WHERE {
    ?obs a sosa:Observation ;
         sosa:resultTime ?time ;
         sosa:hasResult ?r .
    ?r sosa:hasSimpleResult ?value .

    FILTER (
        ?time >= "2025-10-02T00:00:00"^^xsd:dateTime &&
        ?time <= "2025-10-03T00:00:00"^^xsd:dateTime
    )
}
ORDER BY ?time
LIMIT 20
"""
```

Fig 12. Consulta SPARQL por rango de fechas del 2 de octubre al 3 de octubre

Esta consulta nos da las observaciones en un rango de fechas especifioco, en este ejemplo es del 2 de octubre al 3 de octubre

Resultado:

	obs	time	value
1	:obs0289	"2025-10-02T00:00:00"^^xsd:dateTime	"18"^^xsd:float
2	:obs0290	"2025-10-02T00:00:00"^^xsd:dateTime	"47.14"^^xsd:float
3	:obs0291	"2025-10-02T00:10:00"^^xsd:dateTime	"18"^^xsd:float
4	:obs0292	"2025-10-02T00:10:00"^^xsd:dateTime	"48.72"^^xsd:float
5	:obs0293	"2025-10-02T00:20:00"^^xsd:dateTime	"18"^^xsd:float
6	:obs0294	"2025-10-02T00:20:00"^^xsd:dateTime	"60.69"^^xsd:float
7	:obs0295	"2025-10-02T00:30:00"^^xsd:dateTime	"18"^^xsd:float
8	:obs0296	"2025-10-02T00:30:00"^^xsd:dateTime	"64.22"^^xsd:float
9	:obs0297	"2025-10-02T00:40:00"^^xsd:dateTime	"18"^^xsd:float
10	:obs0298	"2025-10-02T00:40:00"^^xsd:dateTime	"63.35"^^xsd:float

Fig 13. Resultado de Consulta SPARQL por rango de fechas del 2 de octubre al 3 de octubre

8. Resultados Obtenidos

Aquí describes de forma breve los hallazgos:

- La temperatura máxima registrada fue de **23.95 °C**.
- La humedad mínima registrada fue de **40.02 %**.
- El promedio semanal de temperatura fue **20.03 °C**.
- Las consultas por intervalos permitieron analizar tendencias temporales.
- GraphDB respondió correctamente y con tiempos de ejecución muy rápidos debido a la estructura RDF.

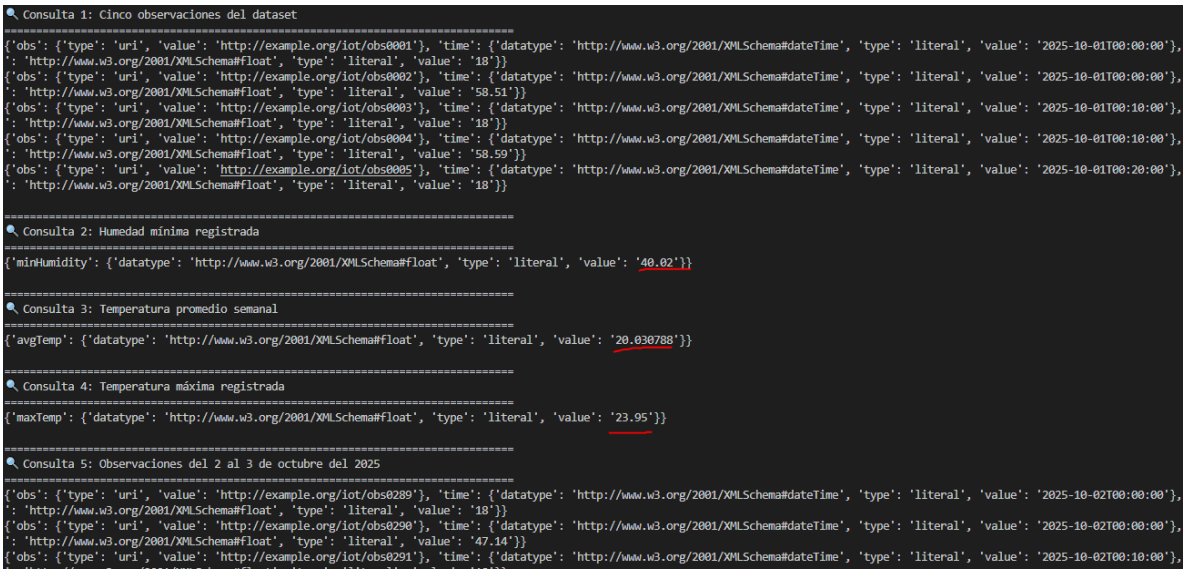

Fig 14. Resultado de correr las consultas desde VS Code

9. Reflexión Crítica

El uso de Web Semántica dentro de un entorno IoT demostró beneficios concretos al momento de analizar los datos del salón de clases. Gracias al modelado con las ontologías SSN/SOSA, fue posible interpretar no solo valores numéricos, sino también su significado dentro del dominio del ambiente interior. Por ejemplo, identificar una temperatura máxima de 23.95 °C o un mínimo de humedad de 40.02 % no solo representa un dato aislado, sino un conocimiento contextualizado dentro de un modelo formal.

Entre las ventajas más destacadas se encuentran la consistencia de los datos, la facilidad para realizar consultas avanzadas (como rangos temporales), y la interoperabilidad, lo cual permitirá agregar más sensores en un futuro sin modificar la estructura general del sistema. Asimismo, el procesamiento semántico facilita la integración con otros grafos, sistemas de monitoreo o aplicaciones inteligentes.

Sin embargo, también se detectan limitaciones. La creación de ontologías y el modelado RDF requiere tiempo, especialmente si se amplía el sistema a más dispositivos. Además, para grandes volúmenes de datos, un triplestore podría requerir más recursos que una base de datos tradicional.

Aun así, el proyecto evidencia que las tecnologías semánticas aportan una capa valiosa de significado, estructura y consulta inteligente a los sistemas IoT. En un escenario real, permitirían construir ambientes inteligentes capaces de analizar variaciones, detectar posibles anomalías y apoyar decisiones basadas en conocimiento, no solo en datos.

10. Conclusión

El análisis semántico del dataset IoT del salón de clases permitió obtener información precisa y estructurada sobre el comportamiento ambiental durante una semana completa. La consulta SPARQL reveló que la temperatura máxima registrada fue de 23.95 °C, lo que indica un ambiente controlado y adecuado para actividades académicas. Del mismo modo, la humedad mínima observada fue de 40.02 %, un valor dentro de rangos saludables y normales para interiores.

El cálculo del promedio semanal de temperatura (20.03 °C) confirma que el salón mantuvo condiciones estables sin variaciones extremas, lo que coincide con un ambiente climatizado o naturalmente regulado. Además, la consulta de intervalos temporales permitió analizar el comportamiento de las mediciones en momentos específicos, dando evidencia de patrones y tendencias durante distintos días.

Estos resultados muestran que la integración entre IoT y Web Semántica mediante SSN/SOSA y GraphDB no solo facilita la interoperabilidad y el análisis estructurado de datos, sino que habilita una capa adicional de inteligencia al permitir consultas complejas sobre información contextual. En conjunto, el sistema demuestra ser un modelo eficiente y escalable para describir y analizar ambientes inteligentes mediante tecnologías semánticas.

11. Referencias

W3C. (2017). *Semantic Sensor Network Ontology (SSN/SOSA)*. https://www.w3.org/TR/vocab-ssn/

Brickley, D., & Guha, R. (2014). *RDF Schema 1.1*. W3C Recommendation. https://www.w3.org/TR/rdf-schema/

Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF*. W3C. https://www.w3.org/TR/rdf-sparql-query/

OpenLink Software. (2023). *SPARQL Tutorial*.

Ontotext. (2023). *GraphDB Documentation*. https://graphdb.ontotext.com/

Haller, A., Janowicz, K., et al. (2017). *The SOSA Ontology: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators.*

12. Github

https://github.com/LuisMario87/Uso-de-Web-Semantica-en-IOT-Temas-Selectos-3